

# 基于图像识别的目标控制与追踪系统设计

章扬

(杭州电子科技大学 电子信息学院, 浙江杭州, 310018)

**摘要:** 本系统采用意法半导体生产的MCU——STM32F407ZET6作为主要控制单片机, 采用OpenMV LLC的OpenMV4 H7作为视觉识别模块来采集彩色图像, 进行优化后的二值化算法后得到黑白图像, 通过边缘检测和矩形检测, 识别矩形框, 之后用以进行图像处理在对采集的图像使用AEC曝光控制技术, 降低快门打开时长以降低图像曝光度, 根据颜色值域的不同识别红色、绿色激光点。根据OpenMV发送的坐标数据, STM32微控制器对数据进行处理, 对比激光点和黑框的位置, 通过PID算法来控制二维云台进行自动跟踪和运动控制。系统采用了SSD和RetinaNet等基于图像处理的目标检测算法, 并结合PID控制算法、插值法等实现了精确的运动和追踪功能。实验结果表明, 该系统能够准确快速地检测和追踪运动目标, 具有良好的实时性和稳定性。

**关键词:** STM32; OpenMV; 运动目标控制; 自动追踪; 图像处理; PID; AEC

DOI:10.16589/j.cnki.cn11-3571/tn.2024.01.024

## 0 引言

随着科技的不断进步和社会的发展, 图像识别技术已经在计算机视觉领域占据了至关重要的地位。图像识别技术的广泛应用领域, 涵盖了从自动驾驶汽车到智能家居安全系统等各个领域, 为实现智能化和自主化提供了强有力的支持。特别是在目标控制与追踪领域, 图像识别技术的应用备受瞩目。通过将图像识别技术与嵌入式系统相融合, 我们可以实现对各种目标的自动识别、跟踪和控制, 从而提高系统的智能程度和自动化水平。本篇论文的主要研究目标在于设计和实施一个基于 STM32 和图像识别技术的目标控制与追踪系统。该系统旨在充分利用先进的图像处理和识别算法, 以实现对目标物体的实时检测、跟踪和控制。通过将 STM32 嵌入式系统与图像识别模块相互融合, 我们得以建构一个高效、低成本的目标控制与追踪解决方案, 适用于各种应用场景, 如智能机器人、自动化仓储系统和监控安全系统等。通过本研究, 我们旨在为图像识别技术在目标控制与追踪领域的应用提供新的思路和方法, 为实现智能化和自主化的系统提供更可行的解决方案。此外, 本篇论文还将为嵌入式系统的开发者和研究者提供关于如何将图像识别技术应用于实际系统的有益信息和经验, 以推动该领域的进一步发展。

## 1 系统整体方案设计

系统整体主要由运动目标控制部分和自动追踪部分构成, 该系统的主控芯片统一使用意法半导体生产的 STM32F407ZET6 单片机, STM32F407ZET6 是意法半导体公司研发的基于 ARM Cortex-M4 内核的 32 位微控制器, 且其支持浮点运算单元 (FPU), 提高浮点数处理性能, 具有更高的运算性能, 可以使用 STMicroelectronics 提供的 STM32CubeMX、STM32CubeIDE 等工具进行开发。基于 ARM Cortex-M4 内核, 时钟频率可高达 168MHz, 具有丰富的通信、定时器和控制外围电路, 充分适用于运动控制与追踪嵌入式系统设计。在系统的视觉方面统一采用

OpenMV LLC 开发的 OpenMV4 H7, 其设备通常配备了高质量的图像传感器, 能够捕捉图像和视频流, 这些传感器可以捕捉彩色图像或者单色 (黑白) 图像。同时 OpenMV 基于 MicroPython 编程环境运行, 它使用一种特殊的单板计算机 (通常基于 ARM Cortex-M 微控制器), 具有内置的图像传感器、处理器和存储器, 可以对图像数据进行快速、准确的处理。该系统通过 MCU 之间的配合将各个模块协同起来工作, 使得对物体的精细控制和对运动物体的精准、快速地追踪<sup>[1,2]</sup>。

### 1.1 运动目标控制模块

运动目标控制模块的设计方案如图 1 所示。在这一方案中, 我们选择了高性能的 STM32F4 作为主控芯片, 以确保系统的稳定性和可靠性。OpenMV 视觉系统被集成进来, 它的主要任务是识别并捕获运动目标的矩形, 并将拐点坐标位置传输给 STM32。一旦 STM32 收到 OpenMV 传来的数据, 它就会进行高效的数据处理, 将视觉数据转化为实际控制指令。这些指令用于控制二维云台, 使其追踪和定位目标, 实现了对运动目标的精确控制。这种整合视觉和控制的方法, 不仅提高了系统的准确性, 还确保了实时性。此外, 基于用户体验的考虑, 通过添加屏幕界面, 用户可以轻松地进行人机交互。他们可以选择不同的运动模式, 如跟随、固定或自定义路径, 以满足不同场景的需求。同时, 用户还能够直接指定目标点, 实现对运动目标的精确控制和定位。这种用户友好的界面大大提高了系统的可用性和便捷性。

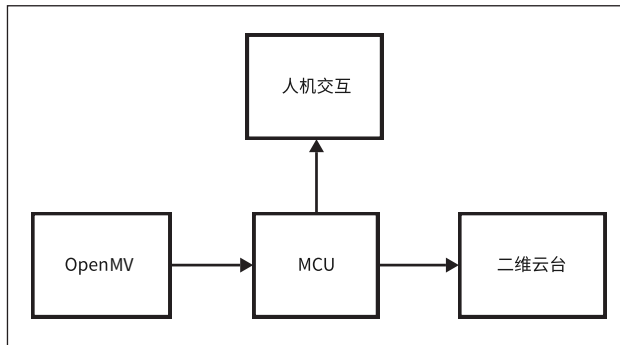


图1 运动目标控制系统

## 1.2 自动追踪模块

自动追踪模块的设计方案如图 2 所示,它采用了一种智能设计,以 STM32F4 作为核心控制芯片,充分发挥其高性能和可靠性。OpenMV 视觉系统起着关键作用,能够精确识别红色和绿色激光点的位置,并将它们的坐标位置传输给 STM32,从而启动数据处理流程。在这个智能系统中,红色激光点被设定为目标位置,而绿色激光点则表示当前位置。STM32 采用 PID (比例-积分-微分) 控制算法,通过处理这两个坐标点的数据,实现了对云台的高度精准控制,确保激光点跟随目标。此外,为了提高用户体验,我们引入了声光提示机制。当系统检测到红色激光点和绿色激光点的坐标差距小于一定范围时,系统会触发声音和光线的提示,以通知用户目标已经在视线范围内。用户还可以通过屏幕进行直观的人机交互。系统提供了一个易于操作的界面,用户可以随时暂停自动追踪,以便进行手动控制或执行其他任务。这个功能增强了系统的灵活性,使用户能够根据具体情况进行操作。

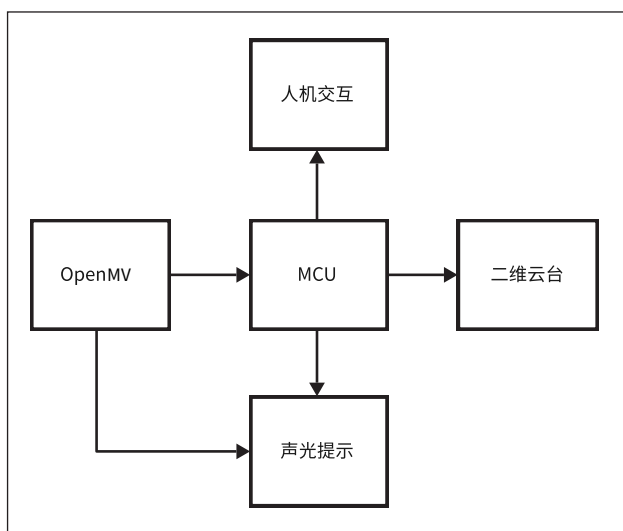


图 2 自动追踪系统

## 1.3 测距模块

测距模块使用了 MyAntenna 激光测距传感器模块,这是一款可见光测距传感器,其原理是相位测距,利用发射光和反射光在空间中传播时发生的相位差来检测距离,精度可以达到毫米级。其中有模块来接收相位差信息,并且可以通过串口将距离信息发送给 MCU。有利于物体的精准控制和快速追踪。

## 1.4 旋转控制模块

旋转控制模块选择了使用  $180^\circ$  角度控制舵机。 $180^\circ$  舵机是通过调节占空比调整旋转角度的器件,输入信号脉冲宽度(周期为 20ms)为 1.5ms 时为  $90^\circ$ , 0.5ms 时为  $0^\circ$ , 2.5ms

时为  $180^\circ$ , 角度与占空比呈线性关系,因此可计算旋转角度。当摄像头传输坐标信息时,可以直接调节角度使激光点复位或巡线,且其调节速度可以通过 PWM 波改变速度而变化,能够更好地调节速度,减小误差。

## 2 系统结构设计

目标控制模块与目标追踪模块都以云台为主要组成部分,两个模块部分结构相同,即将同样的云台结构复刻到两个模块上。主要系统结构是激光笔的定位、云台和舵机的安装再加上摄像头和蜂鸣器等外设的安装与固定共同组成的,完整系统展示如图 3 所示。

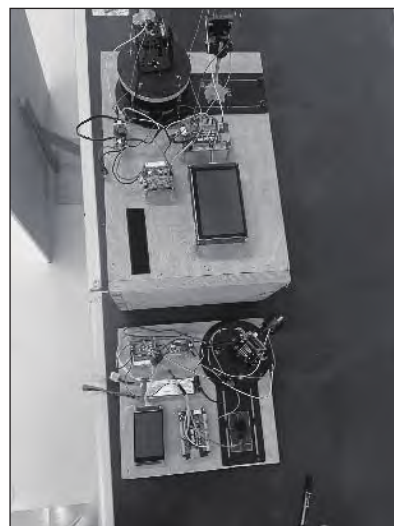


图 3 系统结构图

## 2.1 舵机安装

舵机转向机构在物体的精确控制和迅速跟踪方面扮演着不可或缺的角色。因此,恰当的舵机安装位置和二维云台的机械结构对控制的精确度产生了深远的影响。为了确保舵机的转向顺畅,我们采用了一种高架的安装方式,即在舵机两侧使用铜柱将其提高,并在其两侧增加了加固支撑,以确保结构稳固。此外,我们还在舵机的上下部分安装了机械组件中的铁座,并使用螺丝进行牢固固定,以确保舵机在上下方向上牢固固定。需要特别注意的是,在安装铜柱时,必须确保左右对称性,这对于维持转向的准确性至关重要。此外,为了满足对精细控制的需求,我们需要极力避免舵机的空程误差,这可以通过定期的维护和校准来实现。

## 2.2 OpenMV 位置固定

图片捕获在对运动目标的闭环控制和自动追踪系统的判断中扮演着不可或缺的角色。因此,保证 OpenMV 捕获图片的完整性对系统的控制精度和判断的准确性有着关键的影响。为了保证图像的完整性,我们将运动控制模块的

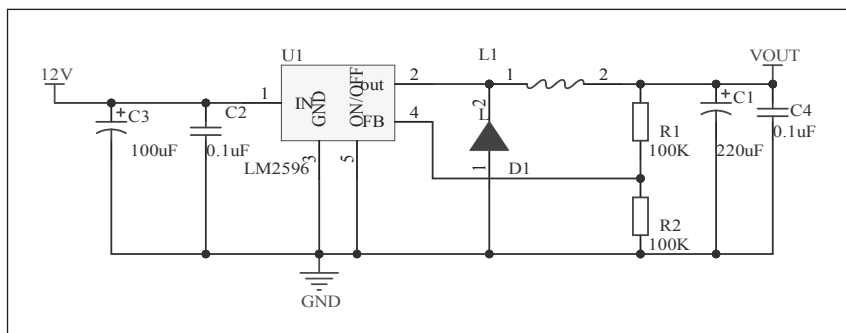


图4 系统部分降压电路

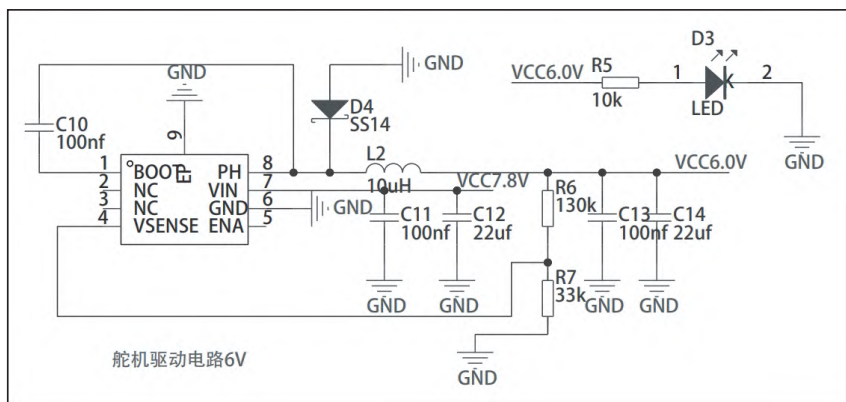


图5 舵机驱动电路

OpenMV 与激光笔平行放置，即垂直面对物体运动的区域：而自动追踪系统的 OpenMV 则放在二维云台上，即与激光笔平行放置于云台上。在此基础上，舵机上下左右扫描的同时，可以保证摄像头的画面一直在不同的角度搜索目标物体，方便对目标物体实施精准追踪。

### 3 系统硬件电路设计

#### ■ 3.1 系统电源设计

系统电源电路如图4所示，本系统使用12V 锂航电池作为供电总电源，使用 Buck 降压电路，通过控制开关管的导通时间比例，通过周期性地输入电压开关切换给负载，以实现输出电压低于输入电压的降压效果。其中5V 供给 OpenMV4 H7 模块，3.3V 作为单片机、OLED 显示屏和声光提示模块的工作。电压变换电路均使用 LM2595S-ADJ 这款芯片，具有噪声低、带载能力强的特点。

#### ■ 3.2 舵机驱动电路设计

舵机驱动电路如图5所示，舵机额定工作电压为6.0V，采用 TI 公司的 TPS5450 降压芯片。单芯片的最大持续输出电流高达5A。输出电压等  $1.229 \times (R6/R7+1)$ ，经过计算选取  $R6=130k\Omega$ ， $R7=33k\Omega$ 。

#### ■ 3.3 声光提示电路设计

使用一个蜂鸣器和一个 LED 灯，并串联一个电阻构成

一个简易的声光提示系统。

## 4 系统软件设计

系统的软件流程图如图7、图8所示，OpenMV 在识别完之后将物体的坐标点发给 STM32 进行处理，运动控制模块的 MCU 在接收到摄像头传回的图像处理结果后，将黑色方框成一百等分，实时判断激光点与黑色方块的位置，通过舵机对激光进行控制，同时可以通过屏幕进行人机交互，选择对应的运动模式。自动追踪模块则先通过 OLED 的人机交互控制摄像头对环境进行扫描，在识别到目标物体点之后对 MCU 发送串口消息，MCU 处理摄像头发送的串口信息，对舵机进行控制，从而实现对目标物体的快速追踪。在追踪成功之后发出声光提示，并在此之后跟随目标物体运动。

### ■ 4.1 图像处理和信

#### 4.1.1 运动控制系统图像处理流程

利用 OpenMV 库实现了一个图像处理流程<sup>[3]</sup>，主要用于检测和分析摄像头捕获的图像中的斑点。首先，它获取了摄像头的一帧图像，并根据条件变量 flag 的值来控制是否执行后续操作。如果 flag 为 0，程序会暂停 200ms 以确保

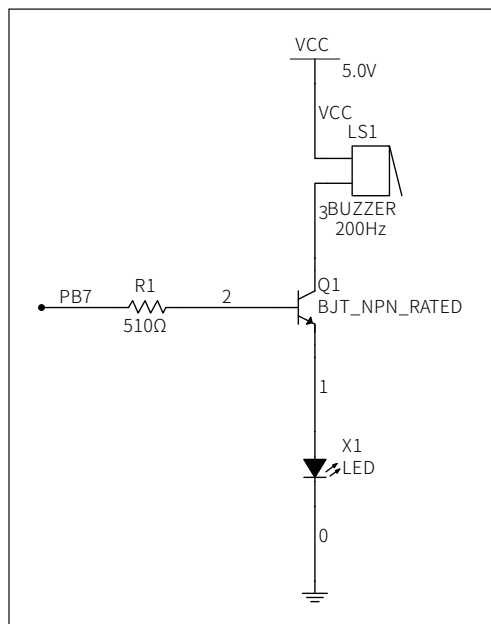


图6 声光提示系统电路

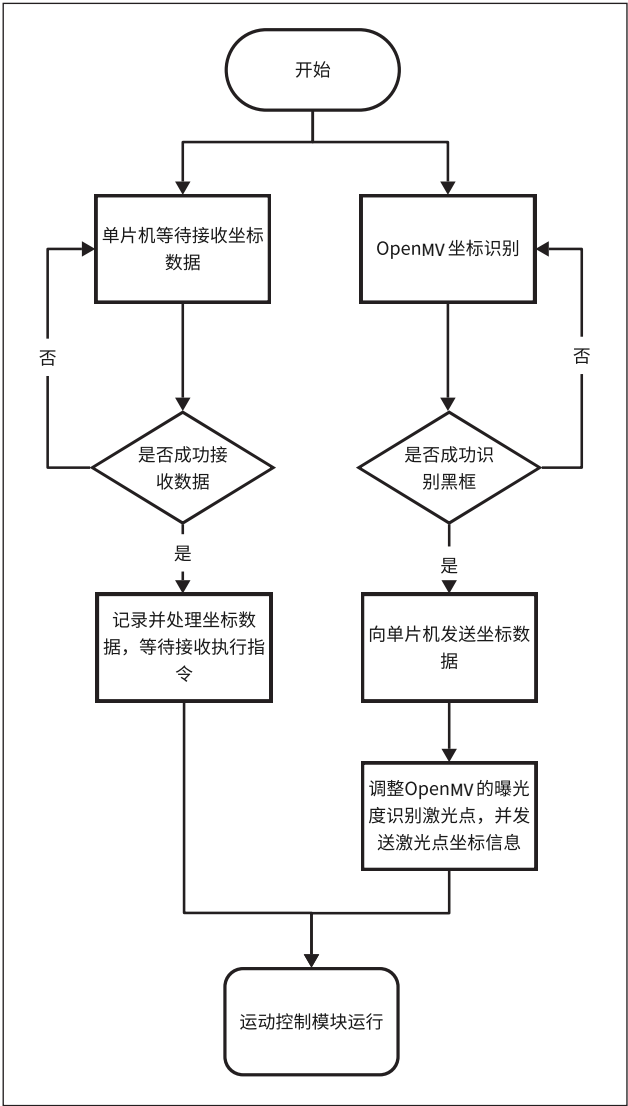


图7 运动控制模块实现流程

图像稳定性。接着，原始图像被绘制在一个灰度图像上，然后应用了侵蚀操作以减小白色区域的大小。代码还定义了一个感兴趣区域（ROI），在该区域内查找斑点，限制斑点的颜色、像素数量和面积。如果发现斑点，代码会绘制它们的边缘和四个角点，并将这些角点的坐标信息打印出来。这段代码的主要目的是检测图像中的斑点，并提取有关它们的信息，这些信息可以在后续的任务中用于目标识别、跟踪或其他图像处理应用中<sup>[4]</sup>。

4.1.2 自动追踪系统图像处理流程

使用 OpenMV 摄像头识别红色和绿色物体的位置，通过串口将它们的坐标传输出去。它通过不断捕获摄

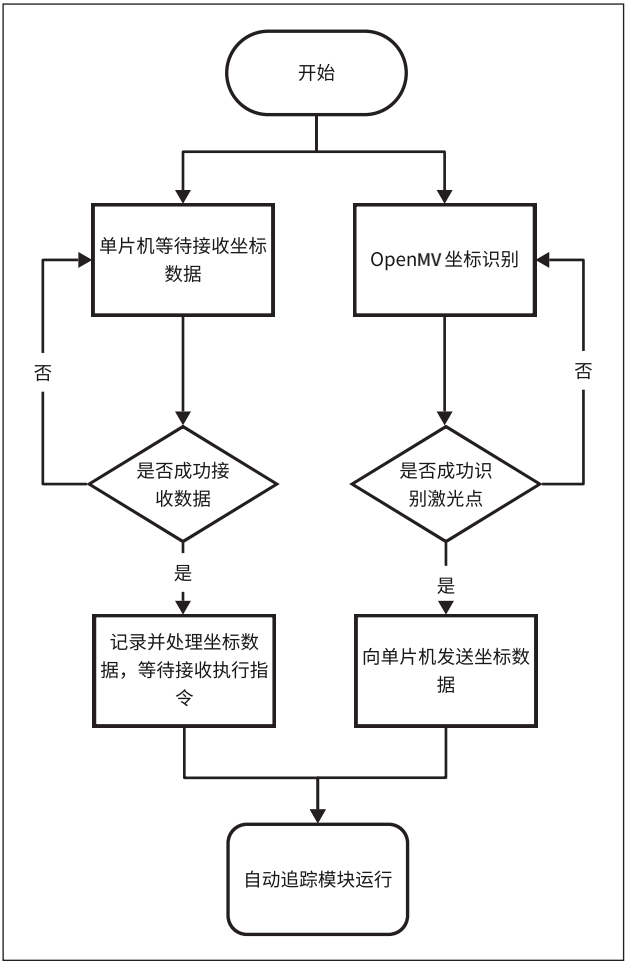


图8 自动追踪模块实现流程

像头图像，交替查找红色和绿色物体的斑点，并发送这些物体的中心坐标，从而实现基本的颜色物体跟踪和控制功能。

4.2 云台控制系统设计

4.2.1 运动控制算法设计

首先通过 UART 接收检查，判断是否接收到完整的数据帧，然后清除 IDLE 标志位，停止 UART 数据的 DMA 传输。接着，它计算接收到的数据的长度，并验证数据是否符合特定条件。

```
img = sensor.snapshot()
if flag == 0:
    sensor.skip_frames(time = 200)
    img_gray.draw_image(img,0,0)
    img_gray.erode(2, threshold = 4)
    rois = (80,40,500,400)
    blobs = img_gray.find_blobs([rois],roi = rois , pixels_threshold=100, area_threshold=100,merge=True)
    if blobs:
        img.draw_edges(blobs[0].min_corners(), thickness=2)
        img.draw_cross ( blobs[0].min_corners()[2][0] , blobs[0].min_corners()[2][1] )
        img.draw_cross ( blobs[0].min_corners()[3][0] , blobs[0].min_corners()[3][1] )
        img.draw_cross ( blobs[0].min_corners()[0][0] , blobs[0].min_corners()[0][1] )
        img.draw_cross ( blobs[0].min_corners()[1][0] , blobs[0].min_corners()[1][1] )
        bottom_left1 = ( blobs[0].min_corners()[2][0] , blobs[0].min_corners()[2][1] )
        bottom_right1 = ( blobs[0].min_corners()[3][0] , blobs[0].min_corners()[3][1] )
        top_right1 = ( blobs[0].min_corners()[0][0] , blobs[0].min_corners()[0][1] )
        top_left1 = ( blobs[0].min_corners()[1][0] , blobs[0].min_corners()[1][1] )
        print(top_left1,top_right1,bottom_left1,bottom_right1)
```

图9 运动控制模块图像处理部分代码

```

while(True):
    clock.tick()
    img = sensor.snapshot()
    if flag == 0:
        blobs = img.find_blobs(red_threshold,x_stride=1, y_stride=1,area_threshold=0, pixels_threshold=0,merge=False,margin=1)
        red_point = color_blob(red_threshold,blobs)
        flag = 1
    elif flag == 1:
        blobss= img.find_blobs(green_threshold,x_stride=1, y_stride=1,area_threshold=0, pixels_threshold=0,merge=False,margin=1)
        green_point = color_blob(green_threshold,blobss)
        flag = 0
    if flag ==0:
        sending_data(red_point[0],red_point[1],green_point[0],green_point[1])
        print(red_point[0],red_point[1],green_point[0],green_point[1])

```

图 10 自动追踪模块图像处理部分代码

```

if (_HAL_UART_GET_FLAG(&huart3,UART_FLAG_IDLE) != RESET)// 通过IDLE标志位判断接收是否结束
{
    HAL_UART_CLEAR_IDLEFLAG(&huart3);//清除标志位
    HAL_UART_DMAStop(&huart3);
    rx_len3 = dma_size - HAL_DMA_GET_COUNTER(&hdma_uart3_rx); //计算出数据长度
    if(rx_len3==13&&rx_buffer3[0]==0xEE&&rx_buffer3[1]==0xDD &&rx_buffer3[12]==0x5b)
    {
        black_rect[0][0]=rx_buffer3[2]*3;
        black_rect[0][1]=rx_buffer3[3]*3;
        black_rect[1][0]=rx_buffer3[4]*3;
        black_rect[1][1]=rx_buffer3[5]*3;
        black_rect[2][0]=rx_buffer3[6]*3;
        black_rect[2][1]=rx_buffer3[7]*3;
        black_rect[3][0]=rx_buffer3[8]*3;
        black_rect[3][1]=rx_buffer3[9]*3;
        red_point[0]=rx_buffer3[10]*3;
        red_point[1]=rx_buffer3[11]*3;
        for(int i=0;i<4;i++){
            for(int j=0;j<4-i-1;j++){
                if(black_rect[j][1]>black_rect[j+1][1]){
                    int temp1,temp2;
                    temp1=black_rect[j][1];
                    black_rect[j][1]=black_rect[j+1][1];
                    black_rect[j+1][1]=temp1;
                    temp2=black_rect[j][0];
                    black_rect[j][0]=black_rect[j+1][0];
                    black_rect[j+1][0]=temp2;
                }
            }
        }
        if(base_3_flag==1){
            x_pid.err = set_x-red_point[0]; // 计算当前速度误差
            x_pid.err_sum+=x_pid.err; //误差累加
            x_pid.out+=
                x_pid.kp * (x_pid.err) //比例P
                + x_pid.ki * x_pid.err_sum //积分I
                + x_pid.kd * (x_pid.err - x_pid.err_pre); //微分D
            x_pid.err_pre = x_pid.err; // 更新上次误差

            if(x_pid.out>300)
                x_pid.out=300;
            if(x_pid.out<-300)
                x_pid.out=-300;
            y_pid.err = set_y-red_point[1]; // 计算当前速度误差
            y_pid.err_sum+=y_pid.err; //误差累加
            y_pid.out+=
                y_pid.kp * (y_pid.err) //比例P
                + y_pid.ki * y_pid.err_sum //积分I
                + y_pid.kd * (y_pid.err - y_pid.err_pre); //微分D
            y_pid.err_pre = y_pid.err; // 更新上次误差

            if(y_pid.out>300)
                y_pid.out=300;
            if(y_pid.out<-300)
                y_pid.out=-300;
            count_x=initial_x-x_pid.out;
            count_y=initial_y+y_pid.out;
            __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_1,count_x);
            __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_2,count_y);
        }
    }
}

```

图 11 目标控制模块部分代码

如果数据验证通过，就从接收缓冲区中提取数据并进行一些排序和处理。接下来，代码计算 PID 控制器的输出，这些控制器用于某种电机控制，根据误差值来更新其输出。最后，它限制 PID 控制器的输出范围，并根据计算的输出值来控制电机或其他设备。最后，代码重新启动 UART 的 DMA 接收以准备接收更多数据。另外，还有一个名为 Order 的函数，负责对某些与 black\_rect 相关的数据进行排序和处理，以及根据特定条

```

x_pid.err = rx_buffer[2]-rx_buffer[4]; // 计算当前误差
x_pid.err_sum+=x_pid.err; //误差累加
x_pid.out+=
    x_pid.kp * (x_pid.err) //比例P
    + x_pid.ki * x_pid.err_sum //积分I
    + x_pid.kd * (x_pid.err - x_pid.err_pre); //微分D
x_pid.err_pre = x_pid.err; // 更新上次误差
if(x_pid.out>300)
    x_pid.out=300;
if(x_pid.out<-300)
    x_pid.out=-300;

```

图 12 自动追踪模块 PID 实现代码

件调整 PID 增益和修改 black\_rect\_divide 数组<sup>[5,6]</sup>。

#### 4.2.2 自动追踪算法

控制舵机在平面扫描左右激光点，在图像中识别到目标物体之后，比较目标物体和追踪激光点的坐标差距，利用 PID 算法，对舵机的运动进行控制，从而达到追踪的效果。

## 5 测试方案与测试结果

测试仪器：卷尺、刻度尺、游标卡尺、秒表。

### ■ 5.1 测试方案

- (1) 首先启动系统，利用屏幕交互移动红色激光点，测试其是否能移动至指定位置。
- (2) 标注正方形框的原点，在任意位置启动复位功能，多次观察能否回到原点，记录偏差。
- (3) 启动基础要求 2，多次测试观察红色激光点绕正方形边框线轨迹，记录绕圈时间 t，在过程中启动暂停功能，并再次启动，检验暂停功能是否完善，同时在暂停时测量与框线偏差。
- (4) 启动基础要求 3，多次测试观察红色激光点绕黑色边框轨迹，记录绕圈时间 t，在过程中启动暂停功能，测量与框线偏差。
- (5) 启动基础要求 4，将黑色边框 A4 纸摆放不同角度，多次测试观察红色激光点绕黑色边框轨迹，记录绕圈时间 t，

表1 复位测试结果表

序号	误差/cm
1	0.36
2	0.54
3	0.21
4	0.33
5	0.64

表2 绕正方形边框线测试结果表

序号	时间/s	最大偏差/cm
1	15.2	0.63
2	15.5	0.81
3	15.2	0.94
4	15.6	0.84
5	15.4	1.13

表3 自定黑色边框A4纸位置巡线测试结果表(采取横放)

序号	时间/s	最大误差/cm	完全脱离黑色框线次数	是否脱离框线移动5cm以上
1	22.61	0	0	否
2	23.32	0	0	否
3	22.55	0.56	2	否
4	22.47	0.23	1	否
5	23.74	0	0	否

表4 任意角度黑色边框A4纸位置巡线测试结果表

序号	时间/s	最大误差/cm	完全脱离黑色框线次数	是否脱离框线移动5cm以上
1	23.81	1.21	2	否
2	23.13	1.53	3	否
3	23.75	1.37	3	否
4	24.17	1.62	2	否
5	24.25	1.29	1	否

表5 完成追踪测试结果表

序号	误差/cm	时间/s	是否有声光提示
1	0	1.21	是
2	0.12	1.23	是
3	0	1.35	是
4	0.20	0.96	是
5	0.13	1.08	是

表6 追踪测试结果表

序号	最大误差/cm	时间/s	是否有声光提示	同时暂停时两点距离/cm
1	0	1.21	是	1.14
2	0.12	1.23	是	1.36
3	0	1.35	是	1.58
4	0.20	0.96	是	2.04
5	0.13	1.08	是	0.76

在过程中启动暂停功能,测量与框线偏差。

(6) 将红色激光复位,启动追踪系统,观察绿色激光在要求位置追踪红色激光点效果,记录追踪成功时间t,测量两点之间偏差。

(7) 红色激光重复上述(4)(5)过程,启动绿色激光追

踪功能,观察运动过程中追踪效果以及声光提示,同时启动两系统暂停功能,测量两点之间偏差。

## 5.2 测试结果与分析

基础要求如表1~表4所示,发挥部分如表5~表6所示。

该系统基本完成题目所需内容,可以实现指定任意点、复位、巡线、暂停并恢复运动、追踪等功能。而由于硬件设施有一定限制,有些许功能有一定误差,其中任意角度摆放黑色框线A4纸误差较大,在算法以及硬件搭建上仍需改进。

## 6 总结

本系统完成了一个,可以根据设定的交互功能设置坐标参数,精确自动控制的运动目标控制系统和自动追踪系统。作品使用常见的STM32微处理器作为主控芯片,并且用OpenMV作为视觉识别,辅助STM32进行控制二维云台。图像部分主要使用霍夫变换、边缘检测和色块检测算法,检测黑色边框或者激光点,同时与STM32拟定通信协议,将必要的坐标信息传输。STM32则使用广用的PID算法对数据进行处理,转换为不同PWM波的值对二维云台进行控制。针对结果,基础部分2误差值不断加大,根据分析,是由于开环造成,因此可以优化算法,现场对正方形框线的四个顶点进行定位,记录四个坐标,与基础部分3一样用PID算法以及插值法形成闭环,提高精确度。若需要实现更高精度的运动与追踪系统,需要采用更高精度的舵机,使用拥有更高图像处理速率的摄像头或识别算法以提高系统相应速度与精确度。

## 参考文献

- \* [1] 赵思雨. 基于HOG特征的跨座式单轨关键部件缺陷识别算法的设计与应用[D]. 西南交通大学,2023-08-05.
- \* [2] 李亚红,贾海龙. 基于混合智能算法的肇事车辆追踪系统设计[J]. 制造业自动化,2013,35(13):3.
- \* [3] 王汝心,马维华. 结合HOG特征的车牌识别方法[J]. 计算机时代,2021(07):1-5.
- \* [4] 柴思博,刘季秋. 基于OpenMV的运动目标预测跟踪云台[J]. 中国科技信息,2021(01):41-42.
- \* [5] 李伟博,李珈毅,张志明等. 基于无剑SoC开源平台的二自由度云台控制系统[J]. 仪表技术与传感器,2022(02):47-51.
- \* [6] 王宁,韩院彬. 低光照强噪声背景下图像多阈值分割方法研究[J]. 计算机仿真,2022,39(03):170-173+178.